Take a Pic, Get a Playlist

AI Recommended Mood–Based Playlists

Renuka Sookdeosingh

Project Reflection

The intention of this project was to create an innovative web application that connects people to personalized music based on their mood. I wanted this to be accessible and easy to understand so anyone could use it. I knew from the start that something like identifying emotion was a very powerful process. It was likely that the results may not be entirely accurate, because emotions are so unique to every individual. However, I thought it was worth a shot, especially with the constantly evolving world of artificial intelligence. Initially, I planned on using Python to code the emotion recognition technology and the rest of the interface. Early on, I realized it would be better to implement existing emotion detection technology and craft my application around it. Though it seemed easy enough, programming an accurate emotion, or even face, recognition script is a project by itself. Plus, I still wanted the chance to incorporate UI/UX and HTML elements in my project.

I turned to the Google Cloud Console to help bring this project to life. With a free trial, I gained access to the Google Vision API and Google App Engine. The Vision API is an artificial intelligence software developed by Google that can detect the key details of an image. It can detect four emotions – joy, sorrow, anger, and surprise. Additionally, it can identify other image properties, such as dominant colors, landmarks, objects, logos, and text. The API provides many opportunities for exploring an image further. Using JavaScript and Node.js, I implemented the code to detect faces in an image. Once it identifies a face, it provides the likelihood of the four emotions. The likelihood ranges from the lowest of very unlikely, unlikely, possible, likely, and the highest of very likely. There is no face recognition involved, only emotion detection.

Take a Pic, Get a Playlist

AI Recommended Mood–Based Playlists

Renuka Sookdeosingh

I coded the interface using HTML and CSS elements. The entire interface is only one dynamic webpage that varies with a specific post request. Using Express for the server, I deployed the application to a website provided by Google Cloud's App Engine. When the user gets to the site, they are prompted to allow camera access. Upon doing so, they can take a picture of themselves right from the site. Once they click the capture button, their image is automatically sent to the Google Vision API to determine the most prominent emotion. Users also have the option to upload a different image with a form, which is also sent to the Vision API with the click of a button. Based on the emotion detected, a designated playlist, embedded from Spotify, shows up next to a preview of the submitted image.

Though the site isn't the nicest to look at, it is clear enough to get the job done. I would have liked to spend more time crafting the page, but I ran out of ideas of what else to add. I knew I didn't want to add anything unnecessary before it cluttered the page or caused the site to run slowly overall. I did spend a lot of time on the mechanics trying to get it to work fluidly. It took a while to get it up and running because there weren't many examples of using the Vision API to create a web application, so I was left doing a lot of trial and error. Using the Express server was also a bit of a pain because it needs certain requirements that aren't very evident.

My first major accomplishment was at least having the form up and running so users could upload an image to get the emotion results and designated playlist. However, I wanted the user to be able to take an image right on the site. I configured the webcam and canvas to do this, but the image needed to be saved to the user's computer automatically. Then, they would have to upload that image to the form. It felt like a user-experience nightmare with excessive steps. Ideally, it should have been that once the user clicks that capture button, their image is sent to Google and their playlist is displayed. It first ended up being an issue with the Express server not sending the data URL of the captured image to Google. Once I got that fixed, the data URL was being sent, but Google was registering it as bad data. I then had to convert the bytes from the data URL to a blob format with more bytes so Google could recognize it. This was one of my major hurdles that took around a month to resolve but getting it working was a relief.

Take a Pic, Get a Playlist

AI Recommended Mood–Based Playlists

Renuka Sookdeosingh

Another issue that took up some time was making sure the correct emotion was being identified in an image. In some instances, the person's emotion was detected as possibly anger, while very likely to be surprised. I realized I needed a better way to iterate through the results and display the corresponding playlists, but it got agitating because the likelihoods are returned as strings and not integers. If it was an integer value, it would be easy to compare, so I needed to convert the strings to integers. I had the results going through some if statements, so if a certain result had a higher likelihood, it would have a higher probability. However, I ran into a problem of the loop finding one emotion that had a high probability of say, possible, and then it would end without going through the other else if statements. If surprise was the first emotion detected and had a likelihood of possible, the program would decide that's the highest probability. It would not check if angry was something higher, such as very likely. This was causing inconsistent results. I eventually resolved this by making all of them if statements instead of else-if. It was a relatively simple fix, but still took up some time to resolve.

I would like to continue developing this application in the future because there are more opportunities to explore. Right now, it correlates a certain emotion to a specific playlist. In further iterations, it could recommend other things based on emotion, including literature, videos, or games. If someone is sad, it can suggest an uplifting video. If they're mad, they can be taken to a relaxing puzzle game. This AI technology is powerful and can be implemented in other programs. I would also like to find other ways to gain a more precise reading of the user's mood. The Google Vision API did provide a relatively easy way to get started. However, it did have some limitations in the types of emotion it can detect and the general likelihood of each emotion. In future iterations, different emotion detection technology could be used. It could hopefully be able to detect more emotions with more accuracy. It may also be helpful to ask the user some additional questions that are more catered to how they're feeling. This way, the program will be less reliant on recognition technology alone. More so, it will help ensure the users have the most personalized experience they can.